ECE 467 Natural Language Processing
Prof Carl Sable
Programming Project 3
Yuecen Wang

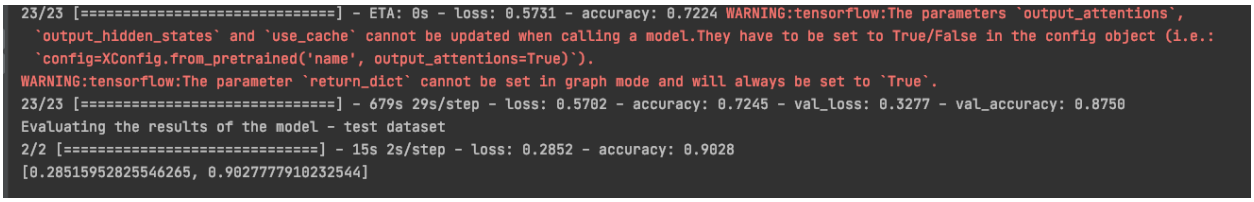
For programming project3, I re-implemented Project #1 – text classification but using deep learning techniques. I have chosen dataset #2 as the dataset for the project. Dataset 2 is a binary dataset with only single (or a few) sentence (s) that generally fits most of the deep learning models. After reading documentations and tutorials for TensorFlow and PyTorch library, I have decided to use TensorFlow as my training library.

During the preprocessing phase of the data, first of all, I have converted the corpus2.labels into a cvs file with the "text" column as each article content for building feature vectors, and the "label" for labeling whether the article is outdoor or indoor. Outdoor label is converted to integer 1, whereas the indoor label is converted to integer 0. The train-test ratio is 0.92, and the train-validation ratio is also 0.92, which makes the training dataset consist of 755 entries, validation dataset has 67 entries, and the test dataset has 72 entries. The information of the dataset can be found in *'./data/corpus2_info.json'* The training, tuning(validation) and test set are split randomly using the train_test_split() function from the sklearn.model_selection library. The tutorial link is https://towardsdatascience.com/best-practices-for-nlp-classification-in-tensorflow-2-0-a5a3d43b7b73

The project consists of two python programs. Main.py and training.py. Main.py is used to build and store raw and clean data efficiently. By using TensorFlow TFRecords for storage, the data can be efficiently used by different models in TensorFlow. A data pipeline is needed to feed my features with labels from the saved filed into the deep learning neural networks. Because of the computing and limited dataset constraints, a deep learning model BERT that has been pre-trained is used and fine-tuning will be conducted with this model. BERT stands for Bidirectional Encoder Representations from Transformers. As a result of using the BERT model, one additional output layer is needed to fine-tune with the model to create state-of-the-art models. Words are converted into numbers through tokenization, with the Hugging Face Transformers framework that can be understood by the BERT model.

I experimented with cleaning the text string, different batch sizes for training, and different learning rate. By writing my own customized string to clean up the text sentences have done a huge improvement for the overall accuracy. On the other hand, change of the batch sizes and learning rate didn't seem to improve the overall accuracy that much. For the final network, I have chosen to use batch size of 32, learning rate of 3e-5 as hyperparameters. In addition to this, I have tried to use different BERT model available. 'bert-base-cased' pre-trained model performs better than the 'bert-base-uncased' pre-trained model for the corpus2 dataset.

As for the result, the training, validation and test accuracy can be seen below:



The validation accuracy is 87.5% with just 1 epoch. The test accuracy is the second number shown in the last list of the figure above. The test dataset has reached 90% of accuracy, which is slightly better than the naiive bayes models I had for the project1.